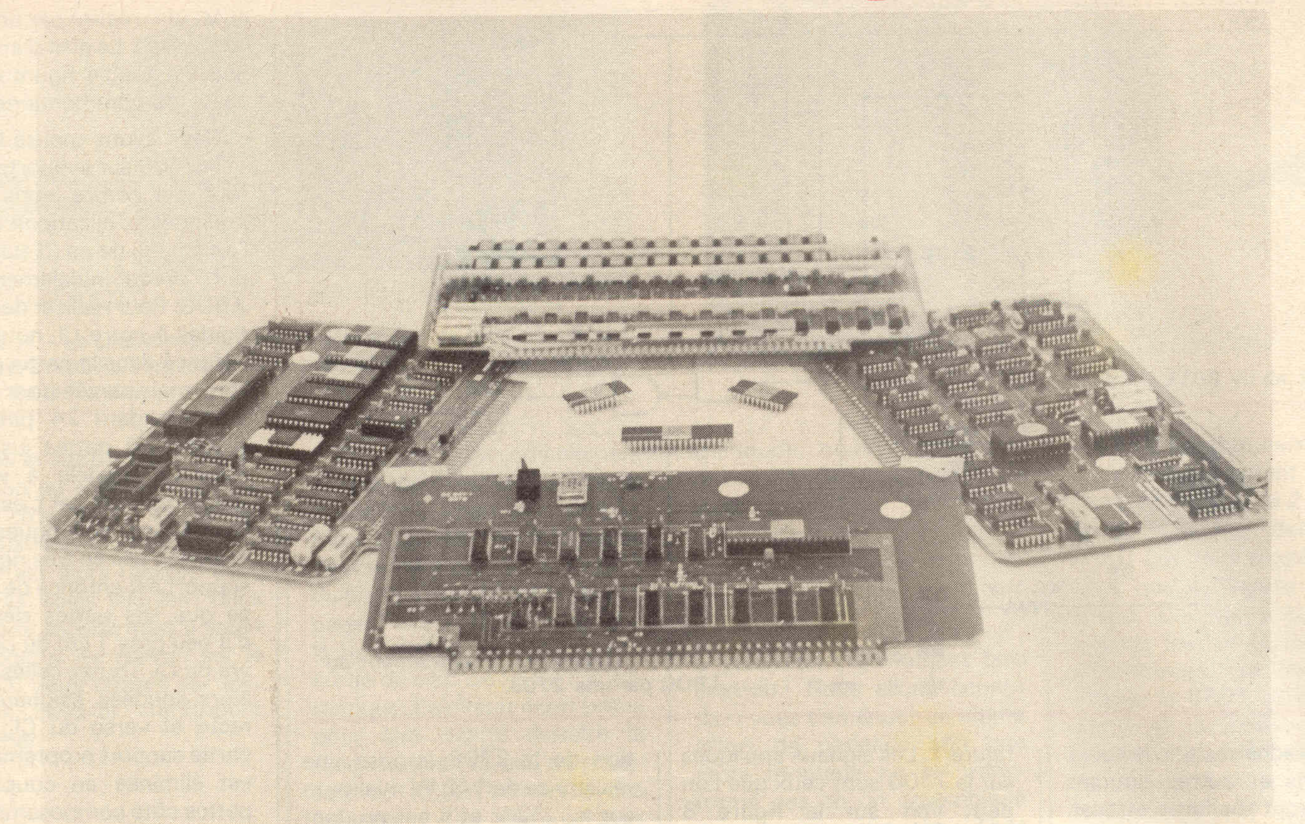
Initiez-vous aux microprocesseurs



en réalisant un mini ordinateur domestique

APRÈS bien des étapes que plusieurs d'entre vous ont jugées un peu longues quoiqu'utiles, nous voici arrivés à l'article tant attendu de la mise en service. Cependant, et compte tenu de ce que nous avons indiqué le mois dernier à propos de la ROM J.BUG, il nous faut réaliser un petit circuit imprimé au tracé simple quoique assez fin ; circuit destiné à supporter la mémoire de remplacement de J.BUG et ses composants connexes. Comme nous serons lancés, nous réaliserons aussi le circuit imprimé de RESET manuel, bien utile lorsque l'on fait ses premiers programmes dont le résultat n'est pas toujours celui escompté. Assez bavardé, passons à l'ouvrage.

Le support de J.BUG bis

La mémoire que nous utilisons pour remplacer J.BUG est une mémoire programmable électriquement et effaçable aux rayons ultraviolets. Le fait qu'elle soit effaçable n'était pas primordial dans ce cas, mais le problème était le suivant :

– il fallait une mémoire facilement programmable par l'auteur de ces lignes,
– il fallait une capacité de 1 k-mots de 8 bits,
– il fallait un temps d'accès inférieur à 450 ns,
– il fallait une mémoire aisément disponible et à un prix relativement bas.

Cette mémoire existe et nous avons indiqué le mois dernier ses diverses références selon le fabricant ; dans la suite du texte nous la baptiserons 2708. Cette 2708 a cependant deux inconvénients :

– Son brochage est totalement différent de celui de J.BUG, ce qui est normal puisque le brochage de J.BUG est unique au monde (mais oui !) ; de plus, cette mémoire ne possède qu'un seul CS (chip select) contre quatre pour J.BUG ; il faut donc prévoir un circuit de décodage d'adresse supplémentaire qui est monté sur le petit CI dont nous avons parlé.
– Il faut trois alimentations à cette mémoire : + 5 V, + 12 V et – 5 V, ce qui nous contraint à monter les alimentations symétriques + et – 12 V et à installer sur notre petit CI un ensemble résistance, zener et chimique pour amener le – 12 V à – 5 V.

Une remarque s'impose à ce sujet pour répondre par anticipation à certains lecteurs ; il existe une version de la 2708 qui ne demande qu'une alimentation 5 V unique (c'est la 2758), cependant elle est plus rare parce que plus récente et beaucoup plus chère ; de plus, l'auteur de cet article n'est pas en mesure de la programmer ; avis aux amateurs !!

Une autre remarque s'impose, cette 2708 étant effaçable aux ultraviolets, la « puce » de silicium se trouve sous une fenêtre en quartz ; c'est très joli à regarder mais une fois que la mémoire est programmée il faut maintenir sur cette fenêtre un morceau

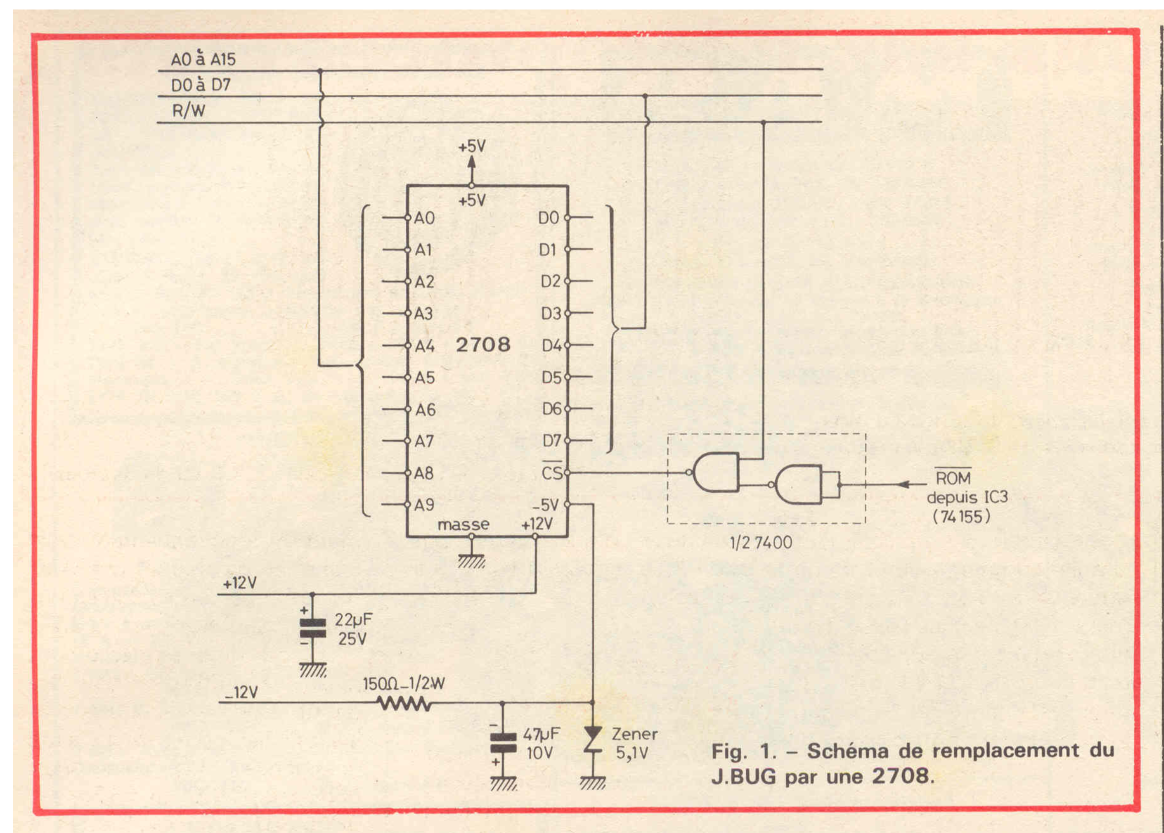


Fig. 1. – Schéma de remplacement du J.BUG par une 2708.

Comme pour tous les CI de cette série d'articles il peut lui aussi être réalisé par une firme spécialisée. Le plan d'implantation est visible figure 4 et se passe de commentaires.

Nous avons monté la 2708 et le 7400 sur supports. Il nous faut, par contre, vous donner quelques explications quant à l'enfichage de ce CI sur le support prévu initialement pour J.BUG. Pour réaliser des pattes rigides à notre CI, nous avons enfoncé dans la partie centrale libre, mais percée pour un support standard 24 pattes, un support 24 pattes justement, mais un modèle à wrapper (c'est impératif car c'est le seul type à avoir de longues pattes à section carrée très rigide). Ce support est enfoncé de façon à ce que ses pattes dépassent d'à peu près 1 cm du côté cuivre du CI. Toutes celles-ci sont alors soudées soigneusement recto et verso du CI, puis la partie support proprement dite est éliminée en coupant les pattes côté composants. Cette petite explication et les photos devraient vous permettre de mener à bien ce petit travail.

Une fois que c'est terminé, s'assurer du bon positionnement des pattes et enficher le

de ruban adhésif sinon le soleil, les néons et autres sources d'UV auront tôt fait d'effacer, ou tout au moins de dégrader, l'information contenue dans la mémoire.

Compte tenu des remarques précédentes, le schéma d'utilisation de la 2708 est indiqué figure 1. Les signaux appliqués sur le 7400 sont ceux que l'on peut voir sur la figure 8 page 210, du n° 1632 ; l'ensemble de la figure 1 se substituant d'ailleurs au boîtier marqué ROM sur cette même figure 8.

Attention ! La consommation de la 2708 impose une résistance de 1/2 W minimum sur le – 12 V et il est prudent d'utiliser une zener de 1 W. Le circuit imprimé (double face étant donné le nombre et le peu de place laissée aux connexions) est visible à l'échelle 1 figures 2 et 3.

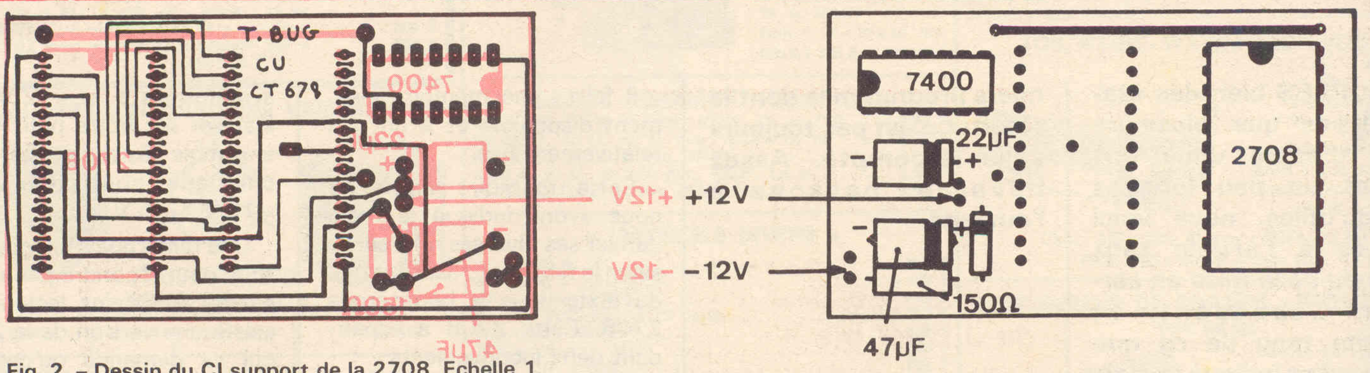


Fig. 2. – Dessin du CI support de la 2708. Echelle 1. Côté cuivre. (Dessin Facim).

Fig. 4. – Implantation des composants sur le support de 2708.

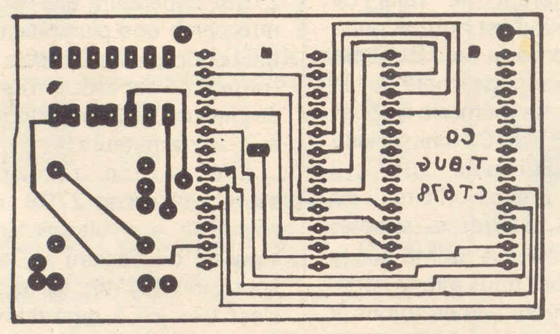


Fig. 3. – Dessin du CI support de 2708. Côté composants. Echelle 1. (Dessin Facim).

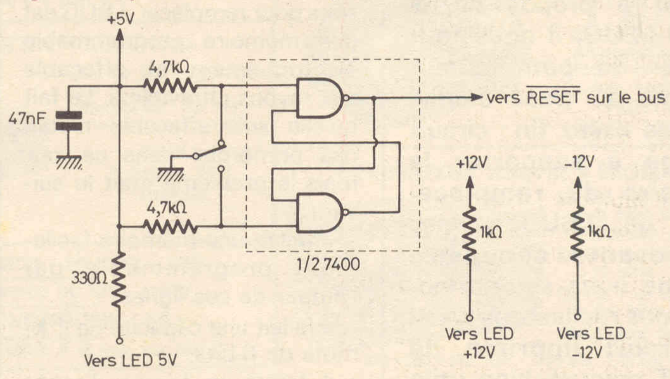


Fig. 5. – Schéma du circuit de RESET manuel.

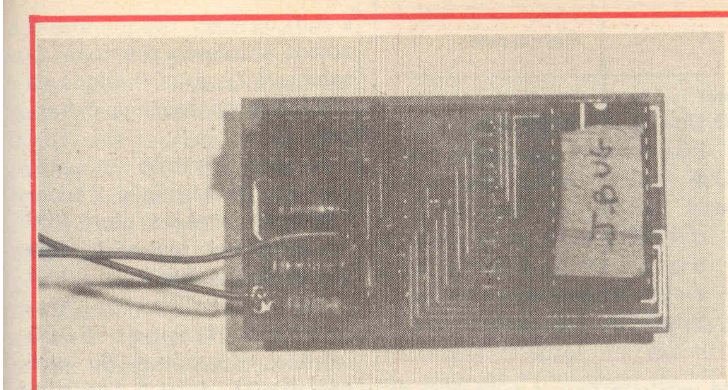


Photo A. – Le circuit support de remplacement de la ROM J.BUG par une 2708. Remarquez le ruban adhésif sur la 2708 pour la protéger de la lumière.

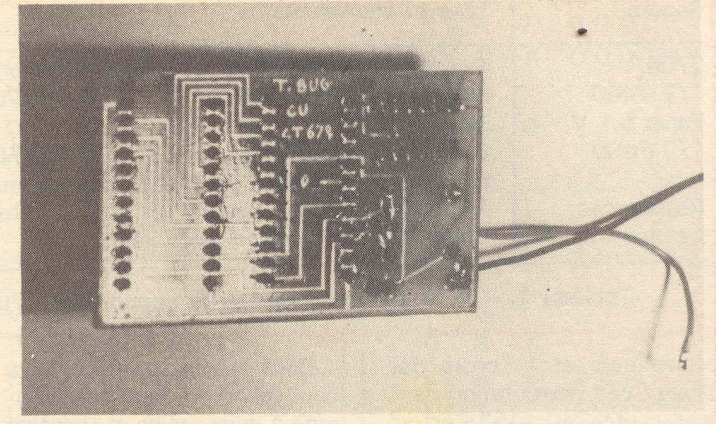


Photo B. – Le support de la 2708 vu de dessous.

CI dans le support de J.BUG ; attention au sens, le 7400 se trouve côté des 6810. La liaison aux alimentations + et – 12 V est alors réalisée avec deux fils isolés souples ; nous les avons soudés mais il serait plus logique de prévoir sur le petit CI, deux cosses sur lesquelles ceux-ci viendraient s'enficher.

Le circuit de RESET manuel

Nous avons vu qu'à la mise sous tension, un circuit spécial de la carte MPU applique sur la patte RESET du 6800 un niveau 0, ce qui a pour effet de positionner certains registres et également de charger le pointeur de programme (PC) avec la valeur contenue dans les cases mémoire d'adresse la plus élevée (FFFE et FFFF). Dans notre cas, cette valeur n'est autre que l'adresse de début du programme J.BUG, ce qui fait qu'après un RESET, le 6800 exécute le programme J.BUG dont nous verrons les conséquences dans ce qui suit. Lorsque l'on met au point des programmes, il arrive souvent que ceux-ci comportent des erreurs qui font faire n'importe quoi au 6800 ; pour pouvoir reprendre le contrôle du système il faut alors exécuter un RESET qui a pour effet (dans notre cas) de relancer le MPU sur J.BUG, et qui nous permet ainsi de rentrer des ordres ou des modifications au programme en cours de mise au point à l'aide du clavier.

Ce circuit est ridiculement simple comme le montre la figure 5 ; il s'agit tout simplement d'une bascule R/S reliée directement à la ligne RESET du bus. La diode 1N914 (fig. 3, page 142, n° 1634) évite que la bascule ne court-circuite la sortie du 555 de RESET automatique. Ce circuit étant prévu pour être monté derrière la face avant (au niveau du poussoir de RESET), nous avons également fixé dessus trois résistances destinées à l'alimentation des 3 LED de contrôle du + 5 V, + 12 V et – 12 V, qui sont également montées sur la face avant.

Le dessin du CI (simple face) est indiqué figure 6, tandis que le plan d'implantation est visible figure 7. Contrairement à l'habitude, nous n'avons pas monté le circuit intégré sur support.

La mise sous tension

Voici enfin le moment tant attendu ! Nous allons cependant vous demander un ultime effort de patience afin que votre montage ne se transforme pas en un assemblage de circuits sans vie. Tout d'abord, avant d'enficher les cartes sur le bus, mesurez les trois tensions, le + et – 12 doivent être exacts à ± 5 % près tandis que le + 5 V doit être dans la fourchette 4,75 V à 5,25 V ; au besoin, ajustez-le avec le potentiomètre prévu à cet effet. Assurez-vous également du fonctionnement de la limitation en courant que vous fixerez à à peu près 2 A en utilisant pour R_{SC} des 0,33 Ω 2 W ; ces deux ampères sont plus que suffisants pour un début. Coupez le courant et insérez les cartes dans leurs connecteurs, non sans avoir enlevé le 6800, le 6820, les 6810 et la 2708. Mettez à nouveau sous tension et vérifiez que le 5 V n'a pas bougé (sinon défaut !), mesurez le – 5 V du circuit support de 2708 et vérifiez l'arrivée du + 12 V, du + 5 V et de ce – 5 V au bon endroit sur le support de la 2708.

Coupez le courant, branchez la carte clavier au moyen de ses deux câbles plats en faisant très attention au sens des

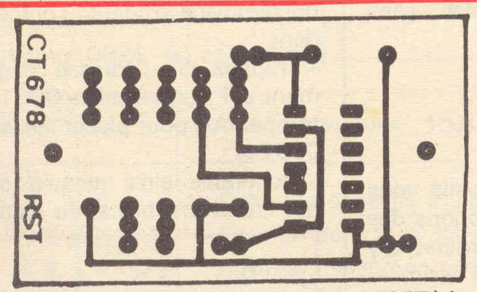


Fig. 6. – Dessin du CI du circuit de RESET (simple face). Echelle 1.

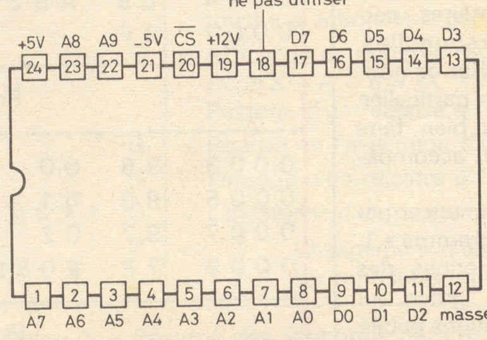


Fig. 9. – Brochage de la 2708, vue de dessus.

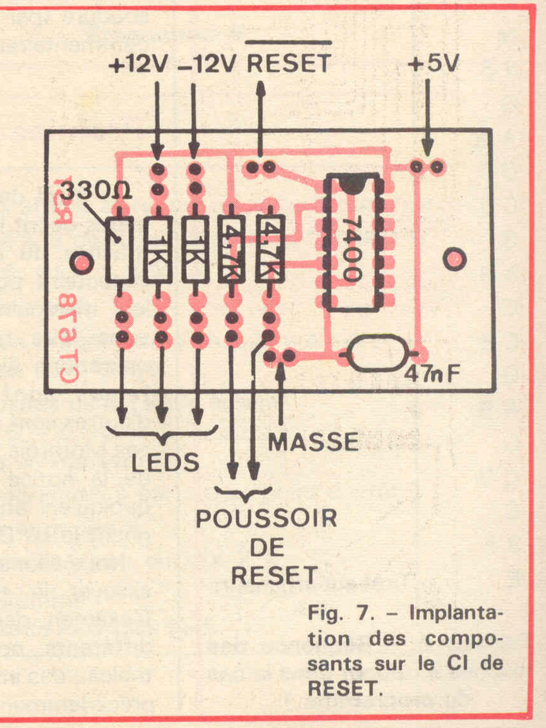


Fig. 7. – Implantation des composants sur le CI de RESET.

Repère	Type et équivalents	Remarques
2708	Types exacts et équivalents publiés le mois dernier	
1/2 7400	SN 7400 N, DM 7400 N, MC 7400 P, SFC 400 E	7400 TTL
Zener 5,1 V	Zener 5,1 V 0,4 W ou mieux 1 W	Tous types
1/2 7400	SN 7400 N, DM 7400 N, MC 7400 P, SFC 400 E	7400 TTL
Inter	Poussoir à 1 circuit 2 positions dont une instable	
Résistances	1/2 W 5 % ou 10 % (les 4,7 kΩ peuvent être des 1/4 W)	
Condensateurs	Chimiques ou plastique selon valeur	

Tableau 1. – Liste des composants du support de 2708 et du RESET manuel.

connecteurs et à l'ordre des câbles ; une permutation dans ceux-ci et le MC 14539 et (ou) le PIA (6820) « passent l'arme à gauche ». Placez sur leurs supports les 6800, 6820, 6810 et 2708 ; attention au sens ! Tous les CI sont dans le même sens sur une carte donnée, sauf le 6800 qui est à l'envers des autres circuits de sa carte. Maintenez un voltmètre sur le + 5 V et mettez à nouveau sous tension ; après 1/2 seconde environ, un tiret doit s'allumer sur l'afficheur de gauche des adresses (afficheur de gauche du groupe de 4). Si tel est le cas, poussez un ouf ! de soulagement, votre mini a 99 % de chance d'être en état de marche.

Séquence des touches	Remarque
RESET	Tiret sur afficheurs
0 0 0 0	
M	
8 6	
G	
A A	
G	
C 6	
G	
B B	
G	
C E	
G	
A B	
G	
C D	
G	
3 F	
E	Tiret sur afficheurs

Tableau 2. – Séquence des touches à frapper dans le cas du programme 1.

Dans le cas contraire, il va vous falloir prendre votre courage à deux mains et repasser en revue vos CI, vos soudures, les pistes douteuses des CI (sauf si ce sont des CI du commerce), les éventuels ponts de soudure, surtout côté composants, au niveau des pattes des CI. Le dépannage d'un tel système ne peut se faire qu'avec un bon oscilloscope et avec une bonne dose de réflexion si la panne est vicieuse ; les schémas étant éprouvés, tout ennui ne peut provenir que d'une des causes ci-dessus ou d'un composant défectueux mais ne vous lancez pas trop vite dans cette hypothèse (surtout pour ce qui est des circuits de la famille 6800).

Avant de nous écrire, vérifiez bien votre montage car nous ne sommes pas devins et nous avons, de plus, vu quantité de montages « archi vérifiés » par leurs réalisateurs et qui étaient en panne à cause d'un pont de soudure (par exemple)... sans commentaires.

Premier contact

Le mois dernier, nous vous avons décrit les fonctions des touches du mini ordinateur ; cependant, pour bien assimiler leur manipulation nous allons vous faire réaliser quelques opérations élémentaires, certaines sont personnelles, d'autres sont extraites de notices Motorola et, en particulier, de la notice (très bien faite quoiqu'en anglais) accompagnant le kit D2.

Nous allons commencer par essayer le « programme » 1. L'examen des fonctions des différents ordres, grâce au tableau des instructions publié précédemment, nous permet de voir que l'on charge l'accu A avec AA, l'accu B avec BB et l'index avec ABCD puis l'on arrive sur une instruction 3 F qui est une interruption programmée (software interrupt). Cette instruction « arrête » le 6800 et, grâce au programme contenu dans J.BUG, nous permet de visualiser les registres internes du MPU, ce qui nous fera vérifier si les changements escomptés ont été faits.

La disposition du programme est celle adoptée le mois dernier.

Nous allons maintenant charger ce programme dans le mini ; pour cela :

– appuyez sur RESET pour avoir le tiret sur l'afficheur gauche des adresses ;

– frappez 0000 puis M, le contenu de la case mémoire d'adresse 0000 est alors visualisé sur les deux afficheurs de gauche (afficheurs de données), frappez alors 86 qui remplace le contenu précédent ;

– frappez G ; l'adresse augment d'1 ; on visualise 0001 ; frappez AA pour placer AA en 0001 ;

– continuez ainsi jusqu'à ce que tous les blocs de deux « chiffres » du programme 1 soient entrés en mémoire ; le tableau 2 vous indique la séquence de touches à frapper ;

– lorsque l'entrée du programme est terminée, frappez E (E fonction et non pas E chiffre, attention) ; le tiret doit être à nouveau visible.

Pour exécuter ce programme, frappez 0000 (adresse du début du programme 1), puis G ; le programme s'exécute et le 6800 s'arrête sur l'instruction 3 F ; les afficheurs indiquent alors : 3 F - 0007 ; 0007 est l'adresse de l'instruction 3 F sur laquelle nous sommes bloqués ;

– frappez alors G ; on visualise alors le contenu de l'index X qui est bien le ABCD escompté ; frappez G, on visualise A qui contient AA ; frappez encore G, on visualise B qui contient BB ; si vous frappez G à nouveau, vous lisez E_9 qui est le contenu du CC (registre d'état) au moment de l'instruction 3 F ; G à nouveau et l'on voit A078 qui est le contenu du pointeur de pile S (nous verrons le pourquoi de cette valeur plus tard). Si vous continuez à appuyer sur G, vous décrivez indéfiniment le cycle ci-avant. Pour changer de fonction, il faut appuyer sur E (de « escape », s'échapper).

Ce programme très simple étant terminé, nous allons passer à quelque chose qui ressemble plus à un programme, puisqu'il s'agit d'un additionneur, simple pour commencer, puisqu'il ne traite que des mots de 8 bits et qu'en plus il additionne en hexadécimal. Cet

0 0 0 0	8 6	A A	L D A	A	# A A
0 0 0 2	C 6	B B	L D A	B	# B B
0 0 0 4	C E	A B C D	L D X		# A B C D
0 0 0 7	3 F		S W I		

Programme 1

0 0 0 3	9 6	0 0	L D A	A	0 0 0 0
0 0 0 5	9 B	0 1	A D D	A	0 0 0 1
0 0 0 7	9 7	0 2	S T A	A	0 0 0 2
0 0 0 9	7 E	E 0 8 D	J M P		J B U G

Programme 2

additionneur est écrit sous la référence programme 2.

Son examen nous montre que le mot contenu à l'adresse 0000 est ajouté à celui contenu en 0001 et que le résultat est placé en 0002.

Ces trois cases mémoires étant réservées, notre programme sera placé en 0003 ; de plus, pour ne pas terminer sur un 3 F nous utilisons une astuce sous forme d'un 7 E E08 D. 7 E signifie JMP (saut) en E08D et E08D n'est autre que l'adresse de début de J.BUG ; ce qui veut dire qu'après avoir lancé notre programme, celui-ci va s'exécuter puis nous verrons apparaître le tiret, maintenant classique, indiquant que l'on est à nouveau sous le contrôle de J.BUG.

Entrez ce programme comme précédemment mais à partir de 0004 ; changez 0000 et 0001 par deux nombres (hexadécimal) à ajouter ; faites 0003 G puis 0002 M pour visualiser le résultat.

La pratique de l'hexadécimal n'étant pas très agréable, nous allons maintenant essayer un additionneur décimal (ou BCD) dont le « listing » est donné en programme 3. Cet additionneur fait appel à une instruction spéciale du 6800 qui est DAA (decimal adjust A) et qui a pour but de traiter le contenu de A par groupe de 4 bits en réalisant la conversion hexadécimal → décimal.

Comme précédemment, les deux nombres (décimaux) à ajouter sont placés en 0000 et 0001 tandis que le résultat sera en 0002. Le programme peut donc être placé à partir de 0003. Entrez en 0000 et 0001 les deux nombres à ajouter ; ces nombres peuvent être décimaux ou hexadécimaux ; auquel cas ils seront automatiquement convertis en décimaux par le programme pendant l'addition. Frappez 0003 G puis 0002 M pour visualiser le résultat.

Nous vous faisons remarquer que, ces petits programmes étant placés dans les 128 premiers octets de mémoire, nous utilisons systématiquement l'adressage direct (voir numéros précédents pour la définition).

Mise au point d'un programme

La première prise de contact étant faite, nous allons exploiter à fond les possibilités du mini de base en utilisant les points d'arrêt, le pas à pas, etc., dans la mise au point d'un programme que nous avons volontairement faussé. En fait, l'exemple ci-après est extrait du manuel d'utilisation du kit MEK 6800 D2 avec l'aimable autorisation de Motorola. Exemple qui est tellement bien fait que nous n'avons pas cru bon d'en réaliser un autre. Allons-y ! Et tout d'abord commençons par entrer en mémoire le programme 4, dont la fonction est très simple, puisqu'il ajoute (en hexadécimal pour simplifier le programme), les cinq nombres compris entre 0010 et 0014 et qu'il place le résultat obtenu en 0015. Ce programme commence sur 0020 et nous allons maintenant détailler pas à pas sa mise au point :

– entrez le programme comme nous l'avons vu précédemment,

– pressez le bouton E pour avoir le tiret,

– entrez les données en 0010 à 0014 ; pour cet exemple, entrez 01, 02, 03, 04, 05 respectivement en 0010, 0011, 0012, 0013 et 0014,

– frappez E à nouveau pour avoir le tiret,

– frappez 0020 G, ce qui a

```
0003  96  00          LDA  A  0000
0005  9B  01          ADD  A  0001
0007  19              DAA
0008  97  02          STA  A  0002
000A  7E  E08D        JMP     JBUG
```

Programme 3

```
0020  8E  00FF        LDS     #00FF
0023  4F              CLR  A
0024  C6  04          LDA  B  #04
0026  CE  0010        LDX     #0010
0029  AB  00     RET  ADD  A  0,X
002B  08              INX
002C  5A              DEC  B
002D  26  FA          BNE  RET
002F  97  15          STA  A  0015
0031  3F              SWI
```

Programme 4

```
0000  86  03          LDA  A  #03
0002  B7  A00C        STA  A  A00C
0005  86  04          LDA  A  #04
0007  B7  A00D        STA  A  A00D
000A  86  05          LDA  A  #05
000C  B7  A00E        STA  A  A00E
000F  86  06          LDA  A  #06
0011  B7  A00F        STA  A  A00F
0014  86  01          LDA  A  #01
0016  B7  A010        STA  A  A010
0019  86  02          LDA  A  #02
001B  B7  A011        STA  A  A011
001E  7E  E0FE        JMP     OUTDS
```

Programme 5

Touches	Fonction effectuée
E	– Donne le contrôle à J.BUG qui attend alors une nouvelle commande
X X X X M	– Examen de X X X X ; passage à la position suivante en frappant G
X X X X V	– Mise en place d'un point d'arrêt en X X X X
V	– Suppression de tous les points d'arrêt
N	– Exécution d'une instruction d'un programme à chaque pression, touche à actionner à partir d'un point d'arrêt
R	– Affichage du contenu des registres dans l'ordre suivant PC – X – A – B – CC – S – PC – X, etc. Passage d'un registre à l'autre en frappant G
G	– Reprise de l'exécution d'un programme à partir d'un point d'arrêt
G	– Passage d'un registre à l'autre en mode R
X X X X G	– Lancement d'un programme commençant en X X X X
P	– } Changement d'un programme sur ou depuis une cassette (voir plus tard)
L	–

Tableau 3. – Résumé des fonctions de J.BUG.

pour effet de faire tourner le programme qui s'arrête en 0031 sur l'instruction 3 F dont nous avons vu le rôle lors du programme 1,

– frappez E puis 0015 M pour regarder le résultat qui est 0A (10 en décimal), ce qui est faux puisque 01 + 02 + 03 + 04 + 05 = 15, soit 0 F en hexadécimal. Nous allons essayer de trouver ce qui ne va pas; frappez E,

– frappez 0029 V, ceci a pour effet de placer un point d'arrêt en 0029 qui est le début de la boucle réalisant les cinq additions successives,

– frappez E puis 002 F et V pour placer un autre point d'arrêt en 002 F afin de visualiser les résultats intermédiaires,

– frappez E puis 0020 G, cela a pour effet de lancer le programme jusqu'à ce que J.BUG rencontre le premier point d'arrêt; à ce moment-là, le programme s'arrête et les afficheurs indiquent AB-0029; 0029 est le contenu du PC (pointeur de programme) et nous sommes placés automatiquement par J.BUG en mode R (examen des registres); si l'on frappe G on visualise le contenu de X (0010), G à nouveau et l'on a A (00), G nous montre B (04), G encore et c'est le CC (D0), G enfin nous montre le pointeur de pile S (00F8),

– frappez E pour quitter ce

DEBUT

Chargement de X par début de DISBUF (A00C)

Chargement de A par (0,X)
Chargement de X par début de DIGTBL

_Conversion BCD-7 segments par INX et DEC A.
_SCNCNT dans SCNREG
_Délai de 1,5 ms

Dernier chiffre affiché ?

non

oui

Décalage à droite de 1 position de SCNCNT

Scrutation du clavier

Fig. 8. – Organigramme simplifié de OUTDS.

mode d'examen des registres, puis G pour continuer l'exécution du programme à partir de ce point d'arrêt; comme le point d'arrêt que nous avons placé en 0029 est situé dans une boucle du programme, nous allons à nouveau nous arrêter dessus et comme précédemment en utilisant la touche G on pourra visualiser les registres; A contiendra la somme partielle; B aura été diminué de 1, X aura augmenté de 1,

– frappez E puis G pour effectuer un nouveau tour de boucle,

– frappez E puis G pour faire encore un tour,

– frappez E puis G pour faire le 4e tour; oh ! surprise, nous nous arrêtons en 002 F ce qui veut dire que le programme a fini de tourner dans la boucle puisqu'il s'est arrêté au point d'arrêt suivant.

A partir de ce moment-là, l'erreur de programmation est évidente; en effet, nous voulons ajouter cinq nombres et nous ne faisons l'opération que quatre fois. L'examen du listing nous permet de comprendre d'où vient l'erreur; elle est située au niveau du LDA B # 04 qu'il faut, bien sûr, changer en LDA B # 5. Pour cela :

– frappez E puis V, cela a pour effet d'enlever tous les points d'arrêt,

– frappez 0025 puis M; on visualise le 04 du LDA B # 04; frappez alors 05 pour remplacer le 04,

– frappez E puis 0020-G; le programme s'exécute et s'arrête sur le SWI situé à la fin,

– frappez E puis 0015-M pour visualiser le résultat qui est correct maintenant.

Cet ensemble de commandes à frapper était peut-être un peu indigeste à lire, mais nous pensons qu'il montre bien la démarche à suivre pour la mise au point d'un programme. Conjointement à cet exemple, le tableau 2 résume les fonctions et le mode d'emploi des touches de J.BUG.

L'art d'utiliser J.BUG

L'utilisation principale du moniteur J.BUG est, bien sûr, l'interprétation des touches frappées au clavier et l'exécution des commandes correspondantes. Pour cela J.BUG contient certains sous-programmes qu'il est intéressant de connaître car on peut les utiliser dans nos propres programmes; nous allons vous en donner un exemple.

Dans le programme mis au point précédemment, il faut, après l'exécution, frapper un certain nombre de touches pour visualiser le contenu de la mémoire d'adresse 0015 qui contient le résultat. Il serait beaucoup plus agréable que celui-ci soit affiché automatiquement sur les afficheurs; pour cela, nous allons faire appel à un sous-programme nommé (par Motorola) OUT DS.

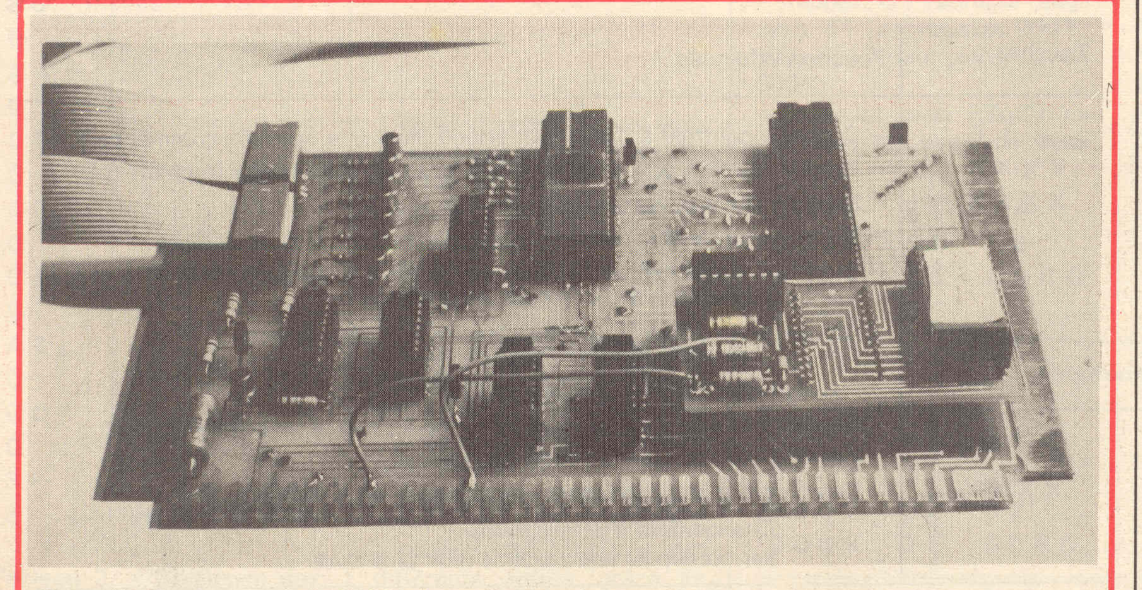


Photo C. – Le support de la 2708 mis en place sur la carte ICAH.

L'organigramme simplifié de ce sous-programme est indiqué figure 8. Pour des problèmes de compréhension utiles dans la suite du texte, nous n'avons pas changé les appellations anglo-saxonnes données par Motorola à certaines mémoires ; celles-ci sont et ont pour signification :

– DISBUF (display buffer = registre d'affichage), ensemble de 8 positions mémoire contenant les « données » à afficher ; les 6 premières positions de DISBUF contiennent les 6 chiffres qui seront visualisés (un chiffre par position mémoire), les deux autres positions servant de stockage temporaire de données. DISBUF commence en A00C.

– DIGTBL est une table (suite de positions mémoire) contenant des données rangées dans un certain ordre afin de réaliser, par programme, la conversion BCD en 7 segments.

– SCNCNT est un mot de 8 bits ne contenant qu'un 1 ; 1 que l'on déplace de gauche à droite pendant OUTDS pour indiquer quel afficheur il faut allumer ; pour cela, on place SCNCNT dans SCNREG.

– DISREG est l'endroit où est placée la valeur à afficher après que la conversion BCD-7 segments ait été effectuée.

L'examen de l'organigramme devient alors relativement simple ; par adressage indexé la première valeur à afficher (contenue au début de DISBUF, donc en A00C) est

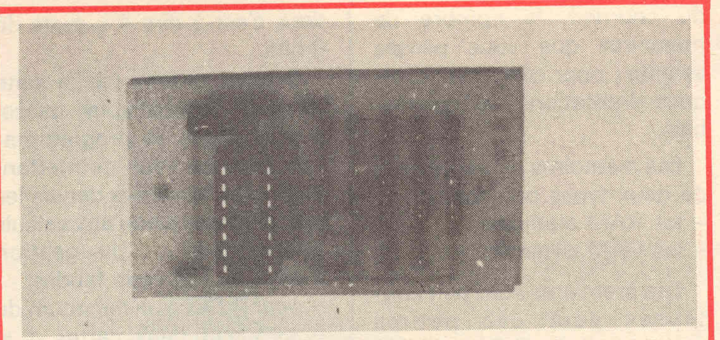


Photo D. – Le circuit de RESET est la simplicité même.

chargée dans l'accu A. X est alors chargé par l'adresse de début de DIGTBL et par une suite de INX (augmentation de X) et de DEC A (diminution de A), la conversion binaire-7 segments s'effectue ; les données placées dans la table DIGTBL l'ont été de telle façon que lorsque A arrive à 0 à la suite des INX et des DEC A, X pointe automatiquement (à cause des INX) sur la case mémoire contenant l'équivalent en 7 segments de ce que contenait A en binaire, ouf !

Cette valeur est alors placée dans DISREG puis SCNCNT est placé dans SCN REG et une boucle de délai de 1,5 ms allume l'afficheur sélectionné par SCNCNT pendant 1,5 ms ; puis SCNCNT est testé ; si l'affichage des 6 chiffres n'est pas fait, le « 1 » qu'il contient est décalé d'une position vers la droite pour allumer l'afficheur suivant en recommençant la boucle que nous venons de décrire.

Lorsque l'affichage des 6 chiffres est terminé, et c'est là le gros inconvénient de J.BUG, le programme part dans le sous-programme de scrutation du clavier. Ceci veut dire que si nous voulons faire un affichage en fin de programme (comme nous l'indiquions au début de ce paragraphe), nous ferons appel à OUTDS (par un JMP E0FE qui est l'adresse de début de OUTDS) ; par contre, si l'on veut faire plusieurs affichages tout au long d'un programme, nous ne pouvons pas utiliser OUTDS puisque cela nous ferait partir dans la scrutation du clavier à chaque fois ; il nous faudra donc recopier les instructions de OUTDS dans notre propre programme (nous vous indiquerons celles-ci en temps utile).

Pour l'instant, vous allez essayer l'affichage grâce au mini programme intitulé programme 5 qui a pour effet (mais c'est évident en le lisant) d'afficher 12-3456.

Attention à la correspondance des positions mémoire de DISBUF et des afficheurs :

– A00C = afficheur d'adresse de poids le plus fort.

– A00D, A00E, A00F = afficheurs d'adresse de poids décroissant, A00F correspondant à l'afficheur de poids le plus faible.

– A010 et A011 respectivement afficheur de données de poids fort et faible.

Exercice

La meilleure façon d'apprendre étant encore de pratiquer, nous vous proposons un petit exercice simple compte tenu de ce que nous avons étudié. Il vous faut écrire un petit programme placé à la suite du programme 4 et qui fera afficher automatiquement le résultat de celui-ci sur les deux afficheurs de données. Attention au piège ! Dans DISBUF il ne faut qu'un chiffre (c'est-à-dire un mot de 4 bits précédé par 4 zéros) par position mémoire ; par exemple, pour afficher 4 sur l'afficheur de poids fort des adresses, il faudra 04 en A00C. Pour afficher 18 sur les deux afficheurs de données il faudra 01 en A010 et 08 en A011. Le résultat du programme 4 (qui se trouve à l'adresse 0015 sous forme d'un mot de 8 bits) doit donc être séparé en deux mots de 8 bits dont les 4 premiers bits sont des 0 (comme dans l'exemple du 18 ci-avant). Bon courage !

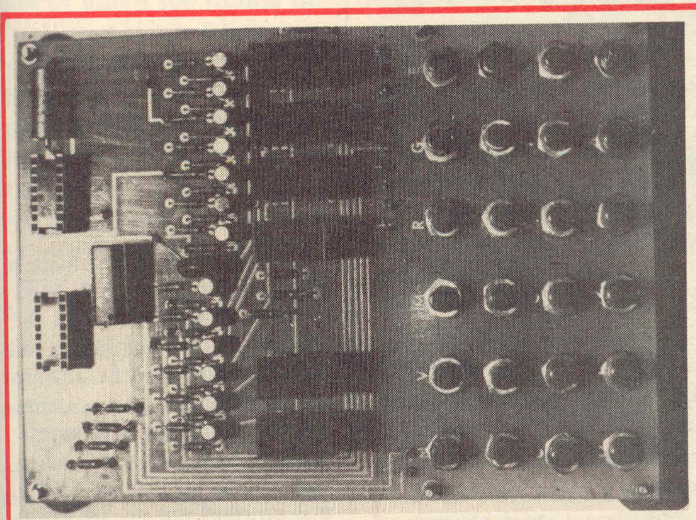


Photo E. – La carte clavier complète vue de dessus...

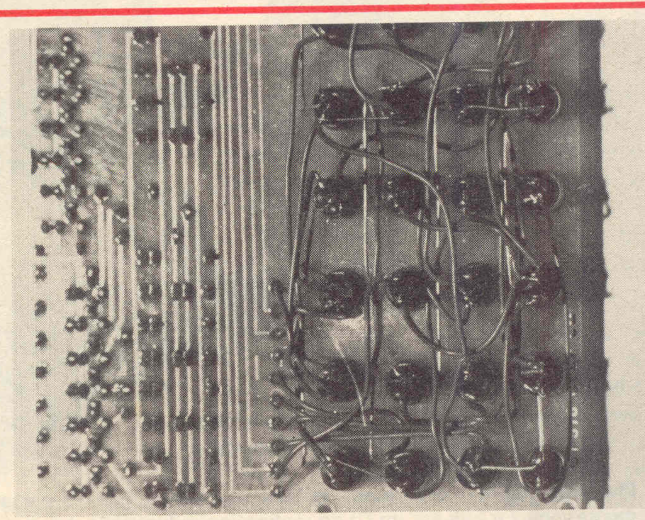


Photo F. – ... et vue de dessous. Attention aux courts-circuits.

Problèmes financiers

Vous allez très rapidement vous rendre compte que notre seule et unique mémoire de 128 mots est bien petite dès que l'on veut faire un programme un tant soit peu complexe; notre premier souci va donc être de réaliser un circuit mémoire de taille plus importante. Malheureusement les boîtiers mémoires, s'ils ne sont pas chers à l'unité, le deviennent très rapidement lorsque l'on désire une mémoire de taille importante à cause du nombre élevé de boîtiers nécessaires.

Pour satisfaire à la demande de plusieurs lecteurs, nous allons tenter une centralisation des commandes de mémoires chez un seul fabricant afin d'avoir des prix de gros (rapport de l'ordre de 2 entre le prix unitaire et le prix par 100); pour cela, il faut que l'auteur de ces lignes sache combien chaque lecteur intéressé pense acheter de mémoires; pour cela, il vous demande d'envoyer sur une feuille (ou carte de visite), sans autre question ou commentaire (pour simplifier le traitement du courrier), le nombre de mémoires que vous pensez acheter; pour cela, voici quelques indications indispensables.

Les mémoires actuelles sont de deux types principaux :
- les RAM statiques,
- les RAM dynamiques.

Les premières sont peu coûteuses pour des petites mémoires (jusqu'à 8 k-mots de 8 bits), simples d'emploi mais ont une consommation non négligeable et une capacité par boîtier assez limitée (1 k-mots de 1 bit par boîtier). Les secondes sont assez complexes d'emploi (circuits de « rafraîchissement » nécessaire), mais reviennent moins cher que les statiques pour des tailles mémoire supérieures à 8 k-mots de 8 bits; de plus elles consomment moins et ont une capacité par boîtier très élevée (16 k-mots de 1 bit par boîtier).

Pour pouvoir vous « amuser » avec le mini ordinateur (bataille navale par exemple), il vous faut au minimum 2 k-mots de 8 bits soient 16 boîtiers de RAM statique 1 k x 1. Cependant la carte support de ces circuits que nous décrivons pourra recevoir jusqu'à 32 boîtiers, c'est-à-dire 4 k-mots de 8 bits.

Si vous désirez par la suite (et c'est souhaitable) passer aux langages de programmation évolués vous permettant toutes les fantaisies depuis les jeux d'échecs jusqu'aux calculs scientifiques ou la gestion d'une usine, il vous faudra :
- pour le Basic un minimum de 8 k, soient deux cartes 4 k, dont nous venons de parler, c'est-à-dire 64 boîtiers de 1 k x 1 ;
- pour le Fortran un minimum de 24 k ; nous réaliserons 20 k en RAM dynamique et il vous faudra donc 4 k en statique, c'est-à-dire 32 boîtiers de 1 k-mots de 1 bit.

Pour l'instant et jusqu'à plus ample informé, nous pensons acheter comme mémoires des 2102 ; ce sont les RAM statiques 1 k x 1 les plus répandues, les moins chères et elles sont rapidement disponibles chez de nombreux constructeurs (Intel, National Semiconductor, Signetics, Texas, etc.).

A titre indicatif, ces mémoires sont vendues 12,00 F par 100 et nous pensons, compte tenu du courrier reçu sur le sujet, dépasser largement ce nombre et essayer d'avoir les prix par 500.

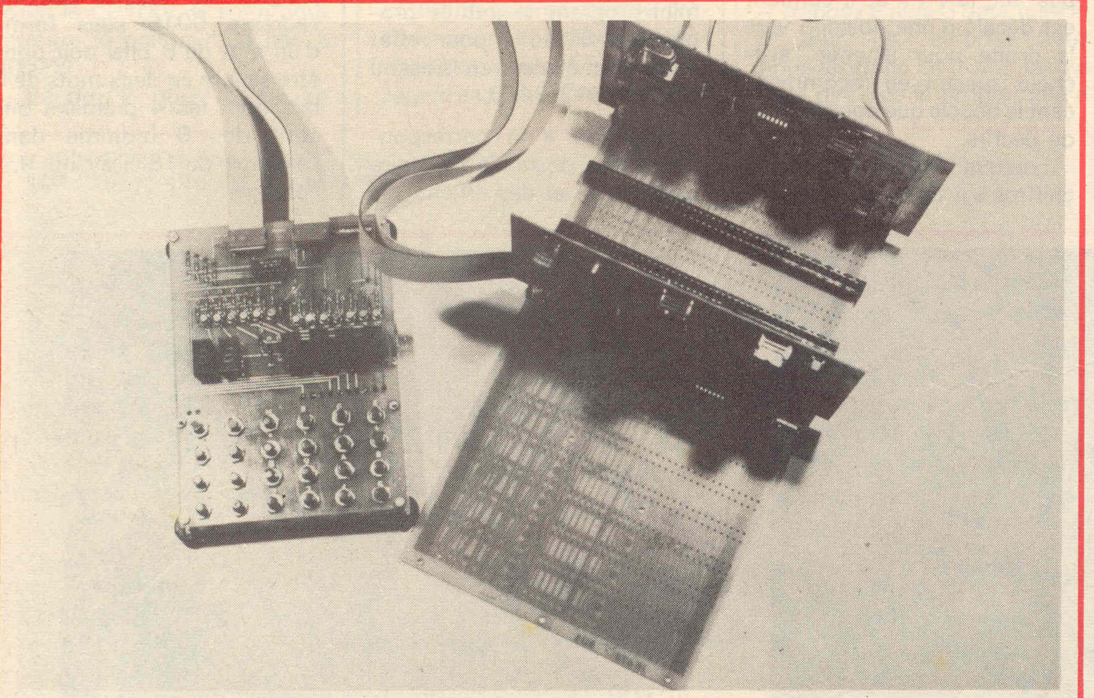


Photo G. – Le mini ordinateur de base en service. Les fils partant de l'arrière du bus vont vers les alimentations. Le bus est un CI de réalisation industrielle, tel que celui que vous pouvez acquérir auprès de la société réalisant les CI de ce mini ordinateur.

En résumé

Adressez à l'auteur sur feuille sans autre commentaire ou question, mais cependant avec votre adresse complète, le nombre de mémoires (statiques pour l'instant puisque nous n'avons parlé que de cela) que vous envisagez d'acheter. Il est bien entendu que ce n'est pas un engagement et que ceci ne cache aucune opération financière ou commerciale quelconque.

D'autre part, indiquez aussi sur la feuille si vous seriez intéressé par la création d'un club 6800 ; club au sein duquel seraient échangés des idées, des schémas, des « combines » et surtout des programmes. Les modalités de fonctionnement de ce club restant bien sûr à définir.

Conclusion

Nous espérons que les quelques exemples que nous avons donnés vous ont un peu familiarisés avec la programmation. Nous vous conseillons d'essayer d'écrire de petits programmes, même sans grand intérêt, uniquement pour vous faire la main.

Bon courage et au mois prochain...

(à suivre)

C. TAVERNIER

Remarques

Deux erreurs sans gravité se sont glissées dans le n° 1633 :
- Page 144, figure 9, il faut échanger les indications « côté cuivre » et « côté composants » ; les lettres du connecteur sont côté composants.
- Page 146, tableau 2, le circuit TTL de la carte MPU (4 ET à 2 entrées) est un 74S08 ou 74LS08 et non un 74508 (qui n'existe pas d'ailleurs).